Evolutionary K-means clustering (E-means) using Genetic Algorithms

1.0

Generated by Doxygen 1.8.5

Fri Aug 14 2015 03:24:52

Contents

Chapter 1

evolutionary-clustering

An enhanced highly parallel K-means clustering algorithm using evolutionary strategies to perform metaheuristic optimization.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 pcg_state_setseq_64 Struct Reference

```
#include <pcg_basic.h>
```

Data Fields

- uint64_t state
- uint64_t inc

4.1.1 Detailed Description

Definition at line 40 of file pcg_basic.h.

4.1.2 Field Documentation

4.1.2.1 uint64_t inc

Definition at line 42 of file pcg_basic.h.

4.1.2.2 uint64_t state

Definition at line 41 of file pcg_basic.h.

The documentation for this struct was generated from the following file:

- include/pcg_basic.h

Chapter 5

File Documentation

5.1 include/cluster.h File Reference

```
#include <gsl/gsl_matrix.h>
#include "pcg_basic.h"
```

Functions

- int lloyd_random (int trials, gsl_matrix ∗data, int n_clusters, gsl_matrix ∗∗clusters, pcg32_random_t ∗rng)
- int lloyd_defined (int trials, gsl_matrix ∗centroids, gsl_matrix ∗data, int n_clusters, gsl_matrix ∗∗clusters)
- int calc_centroids (gsl_matrix ∗centroids, gsl_matrix ∗data, int n_clusters, gsl_matrix ∗∗clusters)
- int calc_bounds (gsl_matrix ∗data, gsl_matrix ∗bounds)
- int random_centroids (gsl_matrix ∗centroids, gsl_matrix ∗bounds, pcg32_random_t ∗rng)

5.1.1 Function Documentation

5.1.1.1 int calc_bounds (gsl_matrix ∗ *data,* gsl_matrix ∗ *bounds*)

Calculates the minimum and maximum bounds based on the data.

Parameters

data	Point to matrix containing the data
bounds	The min/max bounds for each dimensions of the data

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 334 of file cluster.c.

5.1.1.2 int calc_centroids (gsl_matrix ∗ *centroids,* gsl_matrix ∗ *data,* int *n_clusters,* gsl_matrix ∗∗ *clusters*)

Calculate the new centroids using the clustering assignment.

Parameters

centroids	Pointer to matrix containing centroids to be updated
data	Point to matrix containing the data
n_clusters	The number of clusters
cluster	Pointer to vector containing cluster assignment

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 307 of file cluster.c.

5.1.1.3 int lloyd_defined (int *trials*, gsl_matrix $*$ *centroids*, gsl_matrix $*$ *data*, int *n_clusters*, gsl_matrix $**$ *clusters*)

Performs Lloyd's algorithm using the defined centroids.

Parameters

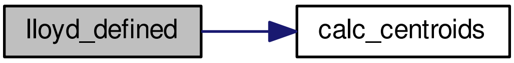
trials	Number of trials to perform
centroids	Pointer to matrix containing the centroids
data	Pointer to matrix containing the data
n_clusters	The number of clusters
clusters	Pointer to array of matrices containing data in clusters

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 191 of file cluster.c.

Here is the call graph for this function:



5.1.1.4 int lloyd_random (int *trials*, gsl_matrix $*$ *data*, int *n_clusters*, gsl_matrix $**$ *clusters*, pcg32_random_t $*$ *rng*)

Performs Lloyd's algorithm using random initial centroids.

Parameters

trials	Number of trials to perform
data	Pointer to matrix containing the data
n_clusters	The number of clusters
clusters	Pointer to array of matrices containing data in clusters

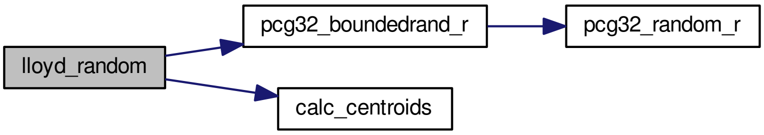
rng	Pointer to the random number generator

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 33 of file cluster.c.

Here is the call graph for this function:



5.1.1.5 int random_centroids (gsl_matrix ∗ *centroids,* gsl_matrix ∗ *bounds,* pcg32_random_t ∗ *rng*)

Generates random centroids within the bounds for each dimension.

Parameters

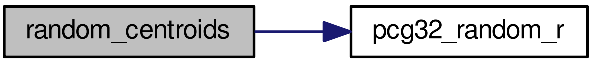
centroids	Pointer to matrix containing centroids to be updated
bounds	The min/max bounds for each dimensions of the data
rng	Pointer to the random number generator

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 362 of file cluster.c.

Here is the call graph for this function:



5.2 include/fitness.h File Reference

```
#include <gsl/gsl_matrix.h>
```

Functions

- double dunn_index (gsl_matrix ∗centroids, int n_clusters, gsl_matrix ∗∗clusters)

5.2.1 Function Documentation

5.2.1.1 double dunn_index (gsl_matrix ∗ *centroids,* int *n_clusters,* gsl_matrix ∗∗ *clusters*)

Calculates the Dunn Index, a metric for evaluating the clustering results.

Parameters

centroids	Pointer to matrix containing the centroids
n_clusters	The number of clusters
clusters	Pointer to array of matrices containing data in clusters

Returns

The Dunn Index

Definition at line 32 of file fitness.c.

5.3 include/io.h File Reference

```
#include <gsl/gsl_matrix.h>
```

Functions

- int load_data (char ∗input, gsl_matrix ∗data)
- int save_results (char ∗output, char ∗output2, char ∗output3, int size, double fitness[size], gsl_matrix ∗∗population, int n_clusters, gsl_matrix ∗∗∗clusters)

5.3.1 Function Documentation

5.3.1.1 int load_data (char ∗ *input,* gsl_matrix ∗ *data*)

Loads the data from as CSV file into a matrix,

Parameters

input	Path to the data file
data	Pointer to the GSL matrix to be populated

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 123 of file io.c.

5.3.1.2 int save_results (char ∗ *output,* char ∗ *output2,* char ∗ *output3,* int *size,* double *fitness[size],* gsl_matrix ∗∗ *population,* int *n_clusters,* gsl_matrix ∗∗∗ *clusters*)

Save the chromosome and fitness value if they are better than previous.

Parameters

output	Path to save the optimal fitness value
output2	Path to save the optimal fitness centroids
output3	Path to save the optimal cluster results
size	Size of the populations
fitness	Pointer to array of fitness values for the population
population	Population of all chromosomes
n_clusters	The number of clusters
clusters	The clusters for each chromosome in the population

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 28 of file io.c.

5.4 include/operators.h File Reference

```
#include <gsl/gsl_matrix.h>
#include "pcg_basic.h"
```

Functions

- void crossover (gsl_matrix ∗parent1, gsl_matrix ∗parent2, pcg32_random_t ∗rng)
- void mutate (gsl_matrix ∗chromosome, gsl_matrix ∗bounds, pcg32_random_t ∗rng)

5.4.1 Function Documentation

5.4.1.1 void crossover (gsl_matrix ∗ *parent1*, gsl_matrix ∗ *parent2*, pcg32_random_t ∗ *rng*)

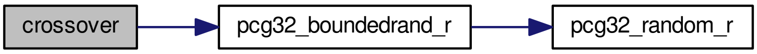
Performs chromosome crossover by randomly selecting a crossover point and randomly either swapping the top or the bottom half.

Parameters

len	The length of the chromosome
parent1	The first parent chromosome
parent2	The second parent chromosome
rng	Pointer to the random number generator

Definition at line 34 of file operators.c.

Here is the call graph for this function:



5.4.1.2 void mutate (gsl_matrix ∗ *chromosome,* gsl_matrix ∗ *bounds,* pcg32_random_t ∗ *rng*)

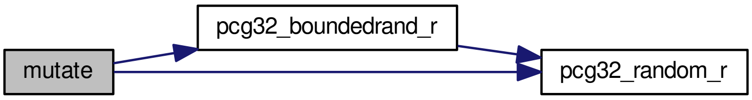
Performs mutation, selects a random row and column in the chromosome and mutates it to a random value within the min/max bounds.

Parameters

chromosome	The chromosome
bounds	The min/max bounds for each dimensions of the data
rng	Pointer to the random number generator

Definition at line 106 of file operators.c.

Here is the call graph for this function:



5.5 include/pcg_basic.h File Reference

```
#include <inttypes.h>
```

Data Structures

- struct pcg_state_setseq_64

Macros

- #define PCG32_INITIALIZER { 0x853c49e6748fea9bULL, 0xda3e39cb94b95bdbULL }

Typedefs

- typedef struct pcg_state_setseq_64 pcg32_random_t

Functions

- void pcg32_srandom (uint64_t initstate, uint64_t initseq)
- void pcg32_srandom_r (pcg32_random_t ∗rng, uint64_t initstate, uint64_t initseq)
- uint32_t pcg32_random (void)
- uint32_t pcg32_random_r (pcg32_random_t ∗rng)
- uint32_t pcg32_boundedrand (uint32_t bound)
- uint32_t pcg32_boundedrand_r (pcg32_random_t ∗rng, uint32_t bound)

5.5.1 Macro Definition Documentation

5.5.1.1 #define PCG32_INITIALIZER { 0x853c49e6748fea9bULL, 0xda3e39cb94b95bdbULL }

Definition at line 49 of file pcg_basic.h.

5.5.2 Typedef Documentation

5.5.2.1 typedef struct pcg_state_setseq_64 pcg32_random_t

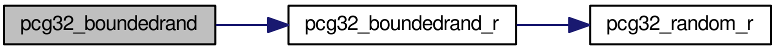
Definition at line 45 of file pcg_basic.h.

5.5.3 Function Documentation

5.5.3.1 uint32_t pcg32_boundedrand (uint32_t *bound*)

Definition at line 112 of file pcg_basic.c.

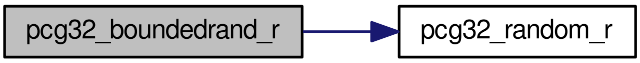
Here is the call graph for this function:



5.5.3.2 uint32_t pcg32_boundedrand_r (pcg32_random_t ∗ *rng*, uint32_t *bound*)

Definition at line 79 of file pcg_basic.c.

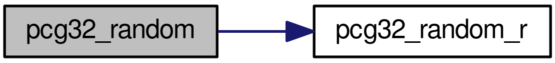
Here is the call graph for this function:



5.5.3.3 uint32_t pcg32_random (void)

Definition at line 69 of file pcg_basic.c.

Here is the call graph for this function:



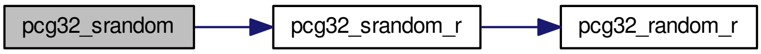
5.5.3.4 uint32_t pcg32_random_r (pcg32_random_t ∗ *rng*)

Definition at line 60 of file pcg_basic.c.

5.5.3.5 void pcg32_srandom (uint64_t *initstate,* uint64_t *initseq*)

Definition at line 51 of file pcg_basic.c.

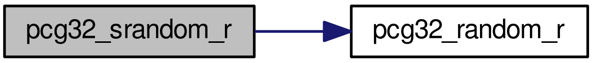
Here is the call graph for this function:



5.5.3.6 void pcg32_srandom_r (pcg32_random_t ∗ *rng,* uint64_t *initstate,* uint64_t *initseq*)

Definition at line 42 of file pcg_basic.c.

Here is the call graph for this function:



5.6 include/selection.h File Reference

```
#include "pcg_basic.h"
```

Functions

- void gen_probability (int size, double fitness[size], double probability[size])
- int select_parent (int size, double probability[size], pcg32_random_t ∗rng)

5.6.1 Function Documentation

5.6.1.1 void gen_probability (int *size,* double *fitness[size],* double *probability[size]*)

Perform the roulette wheel probability selection, an array is populated with the index of the chromosome to select with a frequency based on the fitness value.

Parameters

size	The size of the population
fitness	Pointer to an array of fitness values for population
probability	Pointer to the probability array, populated by function

Definition at line 29 of file selection.c.

5.6.1.2 int select_parent (int *size*, double *probability[size]*, pcg32_random_t ∗ *rng*)

Selects a parent from the population at random with a probability of being selected based on the proabilities provided.

Parameters

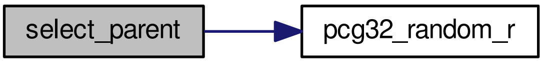
size	The size of the population
probability	Probabilities of each chromosome in population being selected
rng	Pointer to the random number generator

Returns

The index of the parent in the population to select

Definition at line 67 of file selection.c.

Here is the call graph for this function:



5.7 include/utility.h File Reference

Macros

- #define RED "\x1b[31m"
- #define GREEN "\x1b[32m"
- #define YELLOW "\x1b[33m"
- #define BLUE "\x1b[34m"
- #define MAGENTA "\x1b[35m"
- #define CYAN "\x1b[36m"
- #define RESET "\x1b[0m"

Enumerations

- enum debug_code {
  DEBUG_CONFIG = 1, DEBUG_DATA = 2, DEBUG_CLUSTER = 3, DEBUG_BOUNDS = 4,
  DEBUG_CENTROIDS = 5, DEBUG_DUNN = 6, DEBUG_CROSSOVER = 7, DEBUG_MUTATE = 8,
  DEBUG_PROBABILITY = 9 }

  Enumeration of the DEBUG codes.

- enum error_code { SUCCESS = 0, ERROR = 1 }

  Error codes.

Variables

- int DEBUG
- int VERBOSE

5.7.1 Macro Definition Documentation

5.7.1.1 #define BLUE "\x1b[34m"

Definition at line 27 of file utility.h.

5.7.1.2 #define CYAN "\x1b[36m"

Definition at line 29 of file utility.h.

5.7.1.3 #define GREEN "\x1b[32m"

Definition at line 25 of file utility.h.

5.7.1.4 #define MAGENTA "\x1b[35m"

Definition at line 28 of file utility.h.

5.7.1.5 #define RED "\x1b[31m"

Definition at line 24 of file utility.h.

5.7.1.6 #define RESET "\x1b[0m"

Definition at line 30 of file utility.h.

5.7.1.7 #define YELLOW "\x1b[33m"

Definition at line 26 of file utility.h.

5.7.2 Enumeration Type Documentation

5.7.2.1 enum **debug_code**

Enumeration of the DEBUG codes.

Enumerator

DEBUG_CONFIG Print all the values parsed from the config file

DEBUG_DATA Print the contents of the data file

DEBUG_CLUSTER Debug the clutering process using lloyd's

DEBUG_BOUNDS Debug the min/max bounds for each dimension

DEBUG_CENTROIDS Debug the randomly generated initial centroids

DEBUG_DUNN Debug the Dunn Index calculations

DEBUG_CROSSOVER Debug the crossover operator

DEBUG_MUTATE Debug the mutation operator

DEBUG_PROBABILITY Debug output for the probability generation

Definition at line 39 of file utility.h.

5.7.2.2 enum error_code

Error codes.

Enumerator

SUCCESS Successful execution

ERROR Generic error code

Definition at line 56 of file utility.h.

5.7.3 Variable Documentation

5.7.3.1 int DEBUG

Definition at line 40 of file emeans.c.

5.7.3.2 int VERBOSE

Definition at line 40 of file emeans.c.

5.8 README.md File Reference

5.9 src/cluster.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include <stdint.h>
#include <string.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_statistics.h>
#include "utility.h"
#include "pcg_basic.h"
#include "cluster.h"
```

Functions

- int lloyd_random (int trials, gsl_matrix ∗data, int n_clusters, gsl_matrix ∗∗clusters, pcg32_random_t ∗rng)
- int lloyd_defined (int trials, gsl_matrix ∗centroids, gsl_matrix ∗data, int n_clusters, gsl_matrix ∗∗clusters)

- int calc_centroids (gsl_matrix ∗centroids, gsl_matrix ∗data, int n_clusters, gsl_matrix ∗∗clusters)
- int calc_bounds (gsl_matrix ∗data, gsl_matrix ∗bounds)
- int random_centroids (gsl_matrix ∗centroids, gsl_matrix ∗bounds, pcg32_random_t ∗rng)

5.9.1 Function Documentation

5.9.1.1 int calc_bounds (gsl_matrix ∗ *data,* gsl_matrix ∗ *bounds*)

Calculates the minimum and maximum bounds based on the data.

Parameters

data	Point to matrix containing the data
bounds	The min/max bounds for each dimensions of the data

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 334 of file cluster.c.

5.9.1.2 int calc_centroids (gsl_matrix ∗ *centroids,* gsl_matrix ∗ *data,* int *n_clusters,* gsl_matrix ∗∗ *clusters*)

Calculate the new centroids using the clustering assignment.

Parameters

centroids	Pointer to matrix containing centroids to be updated
data	Point to matrix containing the data
n_clusters	The number of clusters
cluster	Pointer to vector containing cluster assignment

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 307 of file cluster.c.

5.9.1.3 int lloyd_defined (int *trials,* gsl_matrix ∗ *centroids,* gsl_matrix ∗ *data,* int *n_clusters,* gsl_matrix ∗∗ *clusters*)

Performs Lloyd's algorithm using the defined centroids.

Parameters

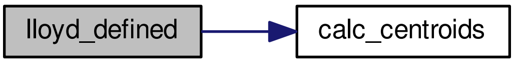
trials	Number of trials to perform
centroids	Pointer to matrix containing the centroids
data	Pointer to matrix containing the data
n_clusters	The number of clusters
clusters	Pointer to array of matrices containing data in clusters

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 191 of file cluster.c.

Here is the call graph for this function:



5.9.1.4 int lloyd_random (int *trials,* gsl_matrix ∗ *data,* int *n_clusters,* gsl_matrix ∗∗ *clusters,* pcg32_random_t ∗ *rng*)

Performs Lloyd's algorithm using random initial centroids.

Parameters

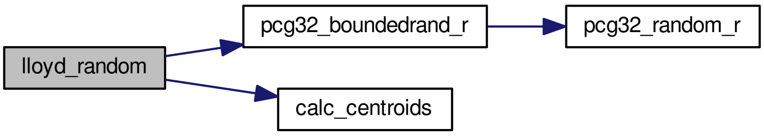
trials	Number of trials to perform
data	Pointer to matrix containing the data
n_clusters	The number of clusters
clusters	Pointer to array of matrices containing data in clusters
rng	Pointer to the random number generator

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 33 of file cluster.c.

Here is the call graph for this function:



5.9.1.5 int random_centroids (gsl_matrix ∗ *centroids,* gsl_matrix ∗ *bounds,* pcg32_random_t ∗ *rng*)

Generates random centroids within the bounds for each dimension.

Parameters

centroids	Pointer to matrix containing centroids to be updated
bounds	The min/max bounds for each dimensions of the data

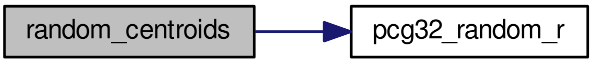
rng	Pointer to the random number generator

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 362 of file cluster.c.

Here is the call graph for this function:



5.10 src/emeans.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <math.h>
#include <unistd.h>
#include <time.h>
#include <confuse.h>
#include <gsl/gsl_matrix.h>
#include "utility.h"
#include "pcg_basic.h"
#include "io.h"
#include "cluster.h"
#include "fitness.h"
#include "operators.h"
#include "selection.h"
```

Functions

- int emeans (void)
- int main (int argc, char ∗argv[])

Variables

- int DEBUG
- int VERBOSE
- int64_t n_clusters = 3
- int64_t trials = 1
- int64_t size = 100
- double m_rate = 0.01

- double c_rate = 0.70
- int64_t max_iter = 10000
- int64_t data_rows = 0
- int64_t data_cols = 0
- char ∗ data_file = NULL
- char ∗ centroids_file = NULL
- char ∗ fitness_file = NULL
- char ∗ cluster_file = NULL
- cfg_opt_t opts [ ]
- cfg_t ∗ cfg

5.10.1 Function Documentation

5.10.1.1 int emeans (void)

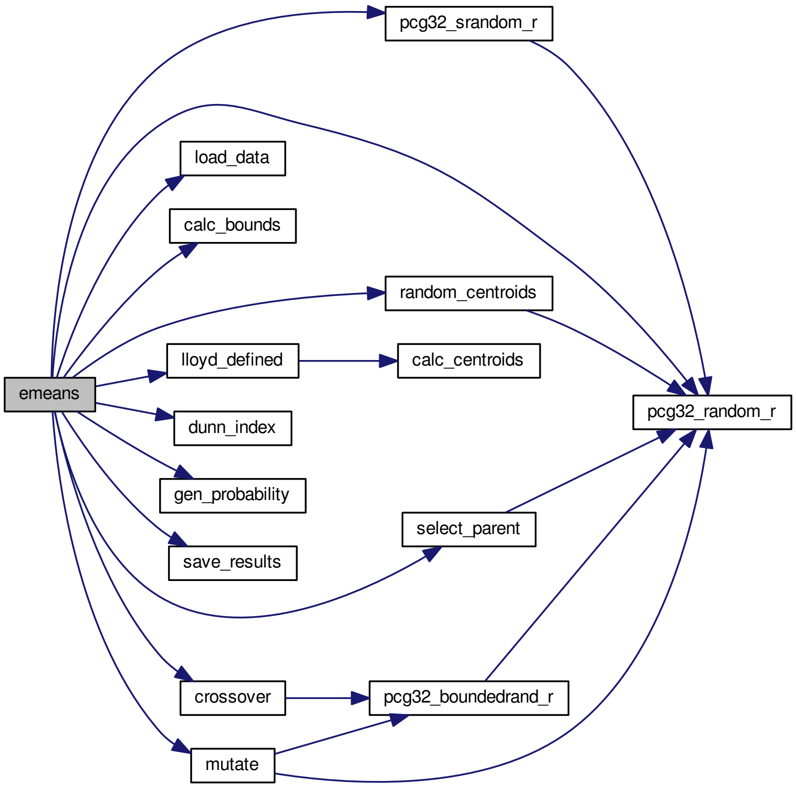
The E-means algorithm, uses a genetic algorithm to optimize the parameters for the K-means implemetation of clustering based Lloyds clustering algorithm.

Returns

Status code, 0 for SUCCESS, 1 for ERROR

Definition at line 81 of file emeans.c.

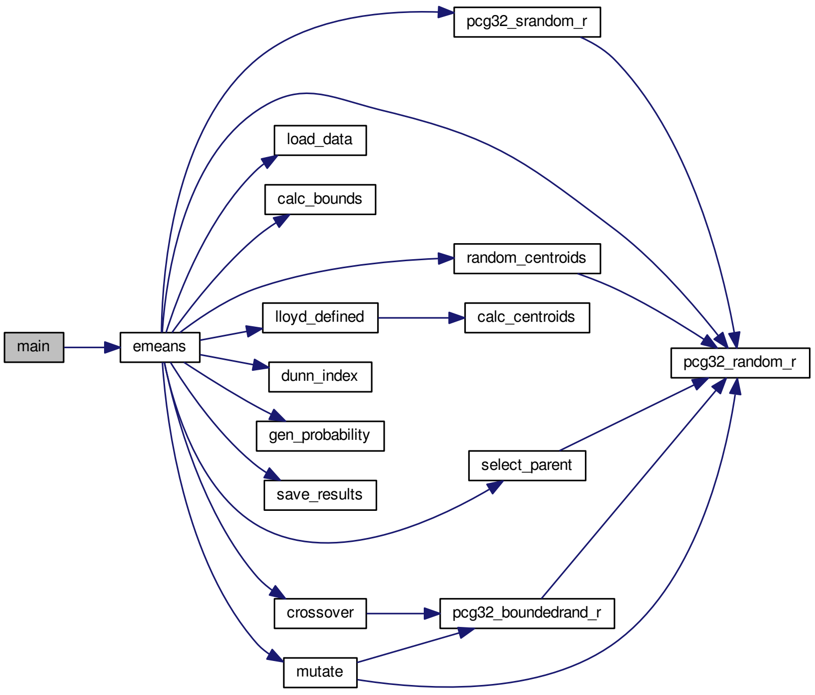
Here is the call graph for this function:



5.10.1.2 int main (int *argc,* char ∗ *argv[]*)

Definition at line 241 of file emeans.c.

Here is the call graph for this function:



5.10.2 Variable Documentation

5.10.2.1 double c_rate = 0.70

Definition at line 47 of file emeans.c.

5.10.2.2 char ∗ centroids_file = NULL

Definition at line 52 of file emeans.c.

5.10.2.3 cfg_t∗ cfg

Definition at line 72 of file emeans.c.

5.10.2.4 char ∗ cluster_file = NULL

Definition at line 54 of file emeans.c.

5.10.2.5 int64_t data_cols = 0

Definition at line 50 of file emeans.c.

5.10.2.6 char∗ data_file = NULL

Definition at line 51 of file emeans.c.

5.10.2.7 int64_t data_rows = 0

Definition at line 49 of file emeans.c.

5.10.2.8 int DEBUG

Definition at line 40 of file emeans.c.

5.10.2.9 char ∗ fitness_file = NULL

Definition at line 53 of file emeans.c.

5.10.2.10 double m_rate = 0.01

Definition at line 46 of file emeans.c.

5.10.2.11 int64_t max_iter = 10000

Definition at line 48 of file emeans.c.

5.10.2.12 int64_t n_clusters = 3

Definition at line 43 of file emeans.c.

5.10.2.13 cfg_opt_t opts[ ]

Initial value:

```
= {
    CFG_SIMPLE_INT("n_clusters", &n_clusters),
    CFG_SIMPLE_INT("trials", &trials),
    CFG_SIMPLE_INT("size", &size),
    CFG_SIMPLE_FLOAT("m_rate", &m_rate),
    CFG_SIMPLE_FLOAT("c_rate", &c_rate),
    CFG_SIMPLE_INT("max_iter", &max_iter),
    CFG_SIMPLE_INT("data_rows", &data_rows),
    CFG_SIMPLE_INT("data_cols", &data_cols),
    CFG_SIMPLE_STR("data_file", &data_file),
    CFG_SIMPLE_STR("centroids_file", &centroids_file),
    CFG_SIMPLE_STR("fitness_file", &fitness_file),
    CFG_SIMPLE_STR("cluster_file", &cluster_file),

}
```

Definition at line 57 of file emeans.c.

5.10.2.14 int64_t size = 100

Definition at line 45 of file emeans.c.

5.10.2.15 int64_t trials = 1

Definition at line 44 of file emeans.c.

5.10.2.16 int VERBOSE

Definition at line 40 of file emeans.c.

5.11 src/fitness.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include <stdint.h>
#include <string.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_statistics.h>
#include "utility.h"
#include "fitness.h"
```

Functions

- double dunn_index (gsl_matrix ∗centroids, int n_clusters, gsl_matrix ∗∗clusters)

5.11.1 Function Documentation

5.11.1.1 double dunn_index (gsl_matrix ∗ *centroids,* int *n_clusters,* gsl_matrix ∗∗ *clusters*)

Calculates the Dunn Index, a metric for evaluating the clustering results.

Parameters

centroids	Pointer to matrix containing the centroids
n_clusters	The number of clusters
clusters	Pointer to array of matrices containing data in clusters

Returns

The Dunn Index

Definition at line 32 of file fitness.c.

5.12 src/io.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <float.h>
#include "utility.h"
#include "io.h"
```

Functions

- int save_results (char ∗output, char ∗output2, char ∗output3, int size, double fitness[size], gsl_matrix ∗∗population, int n_clusters, gsl_matrix ∗∗∗clusters)
- int load_data (char ∗input, gsl_matrix ∗data)

5.12.1 Function Documentation

5.12.1.1 int load_data (char ∗ *input,* gsl_matrix ∗ *data*)

Loads the data from as CSV file into a matrix,

Parameters

input	Path to the data file
data	Pointer to the GSL matrix to be populated

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 123 of file io.c.

5.12.1.2 int save_results (char ∗ *output,* char ∗ *output2,* char ∗ *output3,* int *size,* double *fitness[size],* gsl_matrix ∗∗ *population,* int *n_clusters,* gsl_matrix ∗∗∗ *clusters*)

Save the chromosome and fitness value if they are better than previous.

Parameters

output	Path to save the optimal fitness value
output2	Path to save the optimal fitness centroids
output3	Path to save the optimal cluster results
size	Size of the populations
fitness	Pointer to array of fitness values for the population
population	Population of all chromosomes
n_clusters	The number of clusters
clusters	The clusters for each chromosome in the population

Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 28 of file io.c.

5.13 src/operators.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include <stdint.h>
#include <string.h>
#include <stdbool.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_statistics.h>
#include "utility.h"
#include "fitness.h"
#include "operators.h"
```

Functions

- void crossover (gsl_matrix ∗parent1, gsl_matrix ∗parent2, pcg32_random_t ∗rng)
- void mutate (gsl_matrix ∗chromosome, gsl_matrix ∗bounds, pcg32_random_t ∗rng)

5.13.1 Function Documentation

5.13.1.1 void crossover (gsl_matrix ∗ *parent1,* gsl_matrix ∗ *parent2,* pcg32_random_t ∗ *rng*)

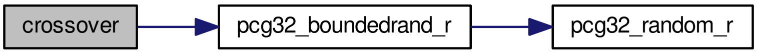
Performs chromosome crossover by randomly selecting a crossover point and randomly either swapping the top or the bottom half.

Parameters

len	The length of the chromosome
parent1	The first parent chromosome
parent2	The second parent chromosome
rng	Pointer to the random number generator

Definition at line 34 of file operators.c.

Here is the call graph for this function:



5.13.1.2 void mutate (gsl_matrix * *chromosome,* gsl_matrix * *bounds,* pcg32_random_t * *rng*)

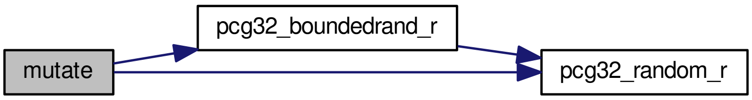
Performs mutation, selects a random row and column in the chromosome and mutates it to a random value within the min/max bounds.

Parameters

chromosome	The chromosome
bounds	The min/max bounds for each dimensions of the data
rng	Pointer to the random number generator

Definition at line 106 of file operators.c.

Here is the call graph for this function:



5.14 src/pcg_basic.c File Reference

```
#include "pcg_basic.h"
```

Functions

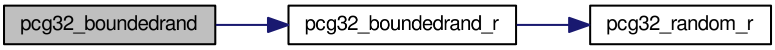
- void pcg32_srandom_r (pcg32_random_t ∗rng, uint64_t initstate, uint64_t initseq)
- void pcg32_srandom (uint64_t seed, uint64_t seq)
- uint32_t pcg32_random_r (pcg32_random_t ∗rng)
- uint32_t pcg32_random ()
- uint32_t pcg32_boundedrand_r (pcg32_random_t ∗rng, uint32_t bound)
- uint32_t pcg32_boundedrand (uint32_t bound)

5.14.1 Function Documentation

5.14.1.1 uint32_t pcg32_boundedrand (uint32_t *bound*)

Definition at line 112 of file pcg_basic.c.

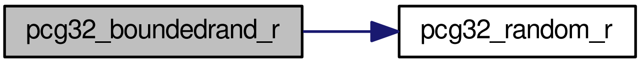
Here is the call graph for this function:



5.14.1.2 uint32_t pcg32_boundedrand_r (pcg32_random_t ∗ *rng,* uint32_t *bound*)

Definition at line 79 of file pcg_basic.c.

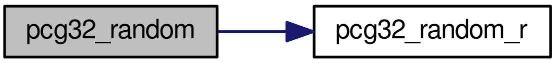
Here is the call graph for this function:



5.14.1.3 uint32_t pcg32_random (void)

Definition at line 69 of file pcg_basic.c.

Here is the call graph for this function:



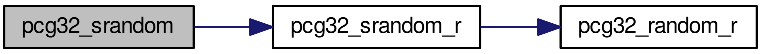
5.14.1.4 uint32_t pcg32_random_r (pcg32_random_t ∗ *rng*)

Definition at line 60 of file pcg_basic.c.

5.14.1.5 void pcg32_srandom (uint64_t *seed,* uint64_t *seq*)

Definition at line 51 of file pcg_basic.c.

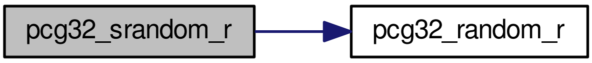
Here is the call graph for this function:



5.14.1.6 void pcg32_srandom_r (pcg32_random_t ∗ *rng,* uint64_t *initstate,* uint64_t *initseq*)

Definition at line 42 of file pcg_basic.c.

Here is the call graph for this function:



5.15 src/selection.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <stdbool.h>
#include "pcg_basic.h"
#include "utility.h"
#include "selection.h"
```

Functions

- void gen_probability (int size, double fitness[size], double probability[size])
- int select_parent (int size, double probability[size], pcg32_random_t ∗rng)

5.15.1 Function Documentation

5.15.1.1 void gen_probability (int *size*, double *fitness[size]*, double *probability[size]*)

Perform the roulette wheel probability selection, an array is populated with the index of the chromosome to select with a frequency based on the fitness value.

Parameters

size	The size of the population
fitness	Pointer to an array of fitness values for population
probability	Pointer to the probability array, populated by function

Definition at line 29 of file selection.c.

5.15.1.2 int select_parent (int *size*, double *probability[size]*, pcg32_random_t ∗ *rng*)

Selects a parent from the population at random with a probability of being selected based on the proabilities provided.

Parameters

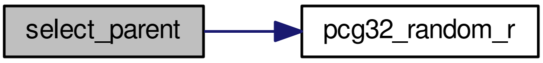
size	The size of the population
probability	Probabilities of each chromosome in population being selected
rng	Pointer to the random number generator

Returns

The index of the parent in the population to select

Definition at line 67 of file selection.c.

Here is the call graph for this function:



Index